

Self Introduction

Portfolio

임베디드SW 개발 지원자 최우영

기술과 열정을 바탕으로 새로운 가치를 창출하는 소프트웨어 엔지니어

Projects

주요 프로젝트

Projects

주요 프로젝트

프로젝트 1

실시간 운영체제 기반 차량용 제어 시스템 개발

- “페달 오조작 방지 보조 시스템”을 주제로 프로젝트 진행



Projects

주요 프로젝트

프로젝트 1

GPIO Driver 사용 방법 분석 및 활용 담당

- RTOS와 autolinux의 GPIO driver 사용법을 분석하고 이를 활용하는 역할을 담당

OUTPUT

```
(void)GPIO_Config(GPIO_GPC(21UL), (GPIO_FUNC(0UL) | GPIO_OUTPUT));
```

- GPIO_GPC(21): C포트 21번 핀. 즉, GPIO_C[21]
- GPIO_FUNC(0UL): 함수 0번으로 설정
- GPIO_OUTPUT: 핀 방향을 출력으로 설정

```
(void)GPIO_Set(GPIO_GPC(21UL), 0UL);
```

- GPIO_C[21] 핀의 값을 0으로 설정

```
(void)GPIO_Set(GPIO_GPC(21UL), 1UL);
```

- GPIO_C[21] 핀의 값을 1으로 설정

5. 원하는 핀 활성화 (여기서는 GPIO_C28번 핀)

```
gpio_request(GPIO_C28, "gpoc");
```

6. 입출력 방향을 출력으로 설정

```
project_tcc_gpio_direction_output(gpio_to_chip(GPIO_C28), 28, 0);
```

7. 핀 On

C ▾

```
project_tcc_gpio_set(gpio_to_chip(GPIO_C28), 28, 1);
```

Projects

주요 프로젝트

프로젝트 1

여러 Task를 하나의 소스 파일에 통합

- 여러 사람이 만든 Task를 git을 이용하여 통합하는 역할

// 생성할 태스크

```
static SALRetCode_t receiveFromPedalTask(void *pArg);
static SALRetCode_t receiveFromFrontBoardTask(void *pArg);
static SALRetCode_t receiveFromRearBoardTask(void *pArg);
static SALRetCode_t sendToBoardTask(void *pArg);
static SALRetCode_t sendToMainCoreTask(void *pArg);
static SALRetCode_t checkAbnormalActionTask(void *pArg);
static SALRetCode_t updateDriveModeTask(void *pArg);
static SALRetCode_t updateAutoholdTask(void *pArg);
```

Projects

주요 프로젝트

프로젝트 2

USB Device Driver 분석 프로젝트

- USB 2.0, 3.0 Specification 분석 → 범위가 방대하여 Architectural Overview 위주로 분석
- Device Driver 구조 분석 → 범위가 방대하여 Mass Storage를 주제로 범위를 제한

Projects

주요 프로젝트

프로젝트 2

USB Specification 분석

- 기능, 인터페이스, 프로토콜, 전력 소비, 성능 지표 분석
- USB driver 분석을 위한 기본 지식을 얻을 수 있었음.

02. 기능

Function

1. Data Transfer 개요

1. Control Transfer
2. Bulk Data Transfer
3. Interrupt Data Transfers
4. Isochronous Data Transfers

Control Transfer

- device가 attached 되었을 때 device를 configure 할 때 사용되는 전송 유형.
- device 고유의 특정 작업을 수행할 때 사용될 수 있음.

Bulk Data Transfer

- 상대적으로 크고 폭발적인 데이터 생성 및 소비에 사용되는 전송 유형.
- 전송 제약 조건에 유연한 것이 특징.

Interrupt Data Transfers

- 즉시성과 신뢰성이 필요할 때 사용되는 전송 유형.
- 사람이 인지할 수 있는 응답성이 필요할 때 사용됨.

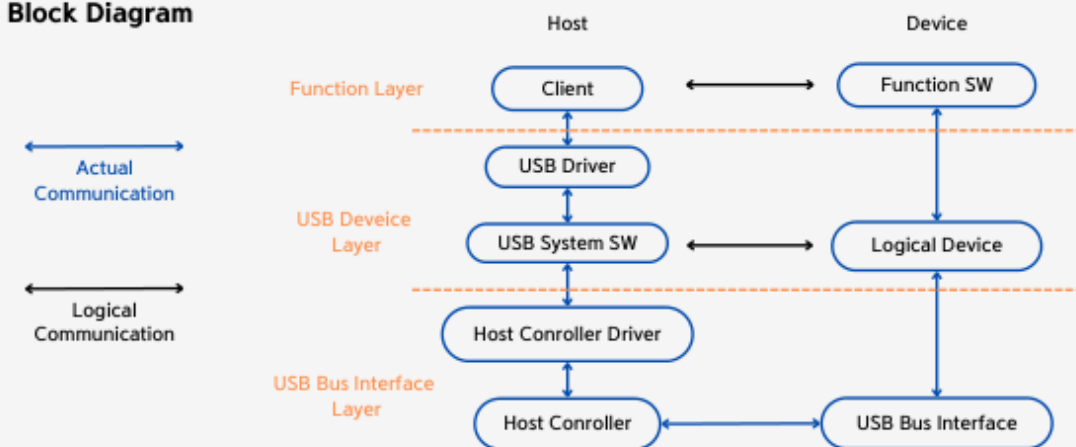
Isochronous Data Transfers

- 사전에 약속된 지연 시간과 대역폭을 점유하는 전송 방식
- "streaming real time transfers"라고도 함.

03. 인터페이스

Interface

2. Block Diagram



04. 프로토콜

Protocol

1. USB 2.0 Bus Protocol 개요

1. Polled Bus, Host Controller initiated
2. Transaction 과정
3. Pipe 모델과 전송 유형
4. Flow control

Polled Bus, Host Controller initiated

- Polled bus란, Host가 주기적으로 traffic flow를 제어하는 방식을 의미.
- USB 2.0은 Host Controller가 모든 데이터 전송을 주도함.

Transaction

- Transaction이란, endpoint에 데이터를 전달하는 것을 의미.
 - transfer: 단일 혹은 다수의 transaction
- token packet, data packet, handshake packet으로 구성.

Pipe 모델과 전송 유형

- pipe: Host와 endpoint 사이의 data 전송 모델.
- pipe에는 stream과 message가 있음

Flow control

- 전송 흐름 제어 방식을 설명하는 부분

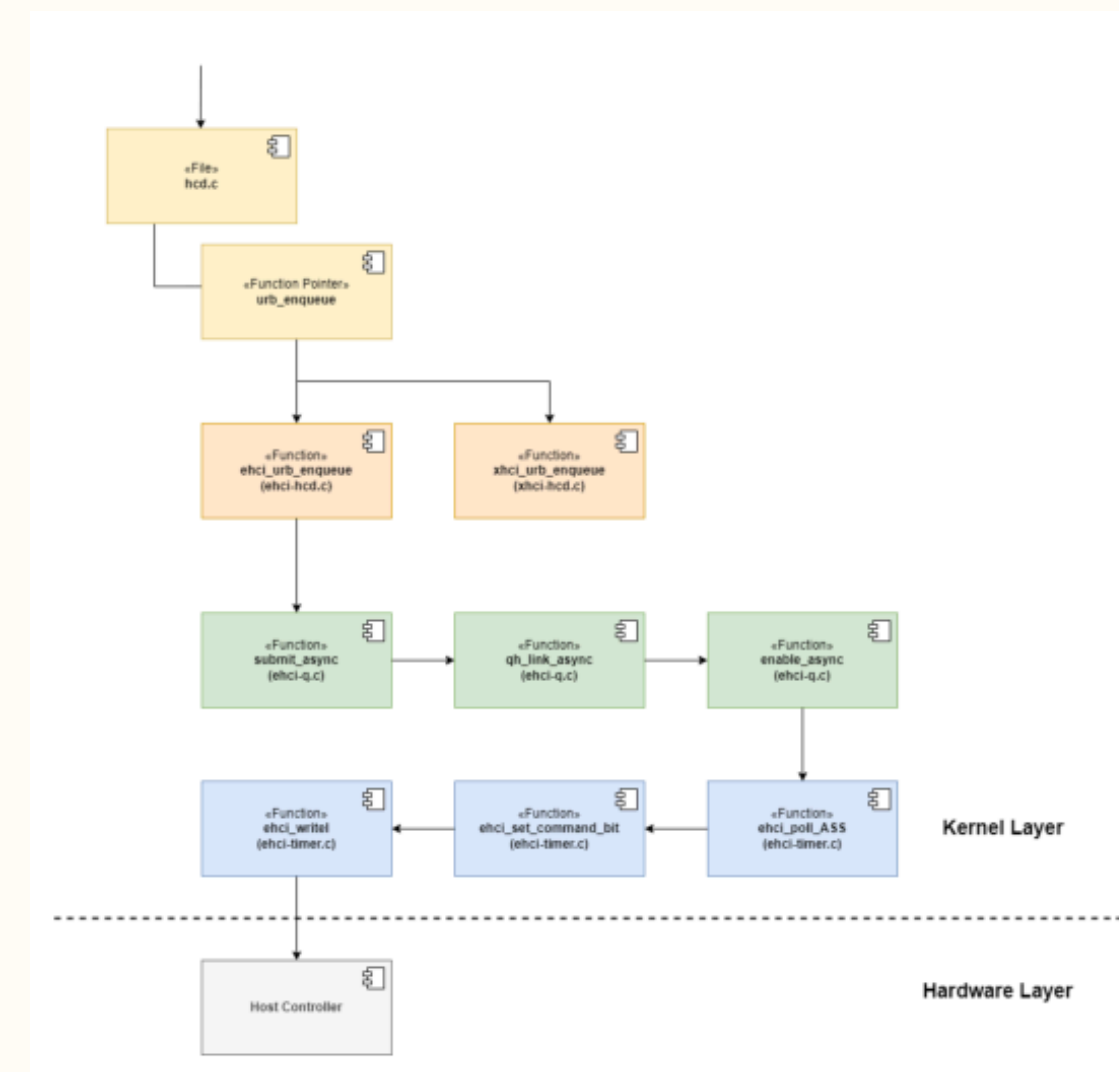
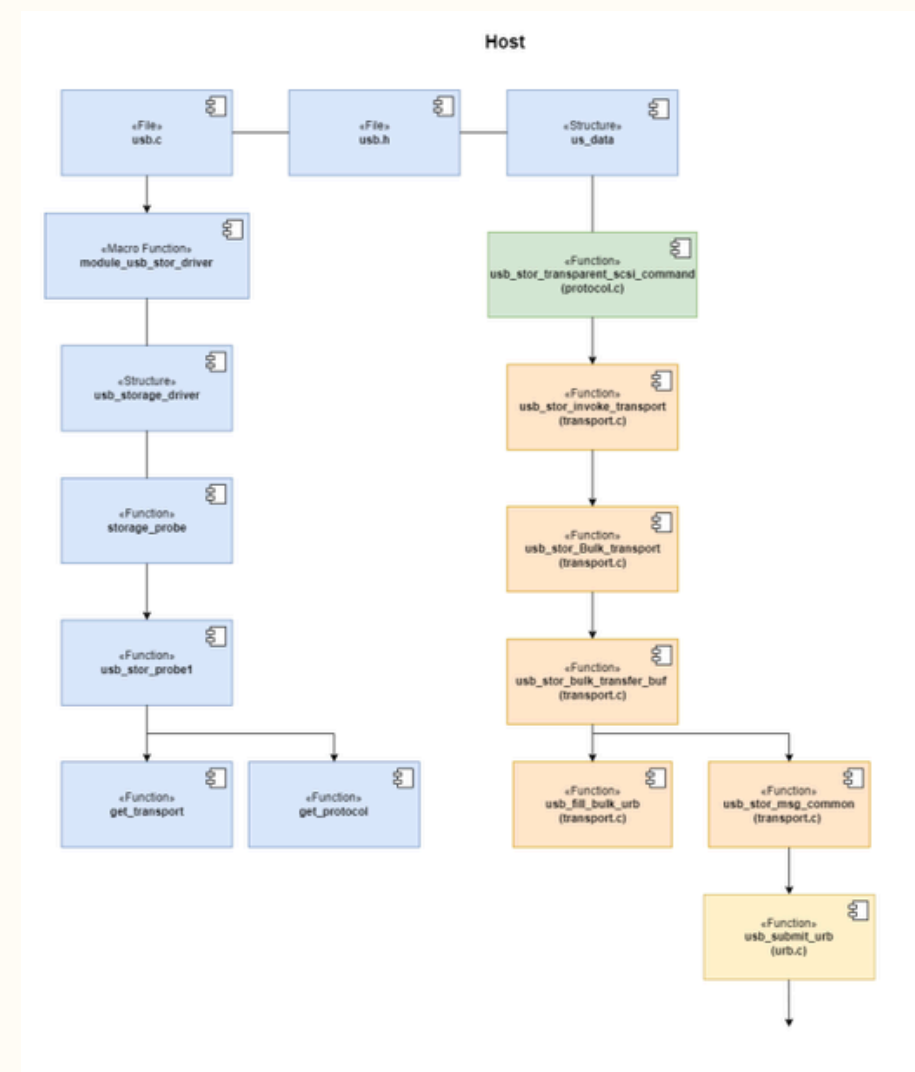
Projects

주요 프로젝트

프로젝트 2

USB Device Driver 분석

- Mass Storage class에 해당하는 소스 파일 선정
- 소스 코드를 보며 모듈의 역할 분석
- 함수 호출을 따라가며 반복적으로 모듈의 역할을 분석
- 상위 level에서 host controller driver 수준까지 데이터 전달 흐름을 파악



Thank you!

감사합니다!

010-6420-2248

uyeongchoe.dev@gmail.com
